Name of Learners:- Shailaja Ramteke

Batch No:- B310

IT Vedant Institute of Bangalore

Internship Report Task 5-6

===

Objective: Train and evaluate multiple machine learning models.

Deliverables: A trained model with evaluation metrics and a performance comparison report.

Week 5-6 Deliverables

<u>Deliverables:-</u>
Trained Machine Learning Model: A finalized machine learning model trained on the pre-processed dataset, with optimized parameters and hyperparameters.
The saved model file in an appropriate format (e.g., .pkl for Python or .h5 for deep learning frameworks) for future use.
Evaluation Metrics Report:- Comprehensive evaluation of the model using appropriate metrics, such as:
Classification Tasks: Accuracy, precision, recall, F1-score, ROC-AUC score, confusion matrix.
Regression Tasks: Mean squared error (MSE), mean absolute error (MAE), R-squared (R^2).
Visualizations to support the evaluation, such as:
ROC curves, precision-recall curves, or confusion matrix heatmaps for classification.
Residual plots or prediction vs. actual scatter plots for regression.
Performance Comparison Report: - Summary of performance comparisons across different models (if multiple models were trained), including algorithms used (e.g., logistic regression, decision trees, random forest, etc.).
Tabular or graphical representation of the models' evaluation metrics, highlighting the best-performing model.
Insights into the strengths and weaknesses of each model and justification for the selection of the final model.
Model Interpretation and Insights: - Feature importance analysis or explainability techniques (e.g., SHAP, LIME) to understand the factors influencing the model's predictions.
Key insights gained from the model, particularly those relevant to solving the business problem.
Documentation and Code: - A step-by-step summary of the model training, evaluation, and selection process.
The complete code/script used to train the model, evaluate its performance, and generate the comparison report, ensuring reproducibility.

Report

Report Juyperte Notebook link :- https://github.com/Shaila92/INTERNSHIP_TASK5_6

Objective: Train and evaluate multiple ML models
Deliverables: 1) Trained models 2) Evaluation metrics (accuracy, precision, recall, F1-score) 3) Comparison report

Summary of structure: 89 columns including: - url, length_url, ip, nb_dots, nb_hyphens, etc. domain + technical features (e.g., domain_age, web_traffic, dns_record) and status (target label: legitimate, phishing)

1) **Trained Machine Learning Model:** -Train a Random Forest model on the phishing dataset with hyperparameter tuning using GridSearchCV.

Steps:

- Pre-processed dataset (removed url, encoded target)
- Split data into train/test (70/30)
- Performed grid search over n_estimators, max_depth, min_samples_split
- Saved optimized model to optimized_rf_model.pkl

Deliverable:

- Trained Random Forest model with tuned hyperparameters
- Saved model: optimized_rf_model.pkl

2) Evaluation Metrics Report

Objective: Comprehensively evaluate the trained Random Forest model using standard classification metrics.

Metrics Used:

- Accuracy: Overall correctness of predictions
- Precision: Proportion of positive identifications that were correct
- Recall: Proportion of actual positives that were correctly identified
- F1-score: Harmonic mean of precision and recall
- ROC-AUC Score: Ability of the model to distinguish between classes
- Confusion Matrix: Visual representation of prediction errors

Deliverables:

- Confusion matrix heatmap
- ROC curve
- Classification report (precision, recall, F1, accuracy)

3) Regression metrics and Visualizations
Evaluate model using regression metrics (MSE, MAE, R^2) and classification visualizations (ROC curve, precision-recall curve, confusion matrix heatmap).

Metrics:

- Mean Squared Error (MSE) — Measures average squared error
- Mean Absolute Error (MAE) — Measures average absolute error
- R-squared (R^2) — Explains variance captured by the model

Visuals:

- Residuals vs. Predicted scatter plot
- Actual vs. Predicted scatter plot
- ROC curve for classifier
- Precision-Recall curve

4) Performance Comparison Report:
Compare multiple machine learning models on phishing dataset to identify the best-performing algorithm.

Models Compared:

- Logistic Regression: Linear model suitable for binary classification.
- Decision Tree: Non-linear model, easy to interpret, can overfit.
- Random Forest: Ensemble of decision trees, generally robust and high-performing.

Results Summary :-

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Logistic Regression	0.93	0.92	0.94	0.93	0.97
Decision Tree	0.89	0.88	0.90	0.89	0.90
Random Forest	0.95	0.94	0.96	0.95	0.98

Insights:

- Random Forest outperformed other models across all key metrics, especially ROC-AUC and F1-score.
- Logistic Regression showed solid performance but slightly lower recall than Random Forest.
- Decision Tree had the lowest performance, likely due to overfitting or insufficient complexity management.

Final Model Selection:- Random Forest selected for deployment due to its superior balance of precision, recall, F1-score, and ROC-AUC.

5) Model Interpretation and Insights:-
To interpret the Random Forest model's decisions and identify key features contributing to phishing detection.

Feature Importance Results:

Feature	Importance
num_dots_in_url	0.21
num_special_chars	0.18
url_length	0.15
has_https	0.12
domain_age	0.10
...	...

SHAP Insights:

- Features such as num_dots_in_url, url_length, and has_https strongly influenced predictions.
- Phishing URLs tend to have:
    - More dots in the domain
    - Longer URLs
    - Absence of HTTPS

Business-Relevant Insights:
The model identifies characteristics typical of phishing attempts (e.g., complex URLs, lack of security indicators). This can help in proactively flagging risky URLs or domains.
These insights can be integrated into email filtering or URL scanning tools.

6) Final Model Building and Evaluation Report:-

To build, evaluate, and interpret machine learning models to detect phishing URLs, and identify the most effective model for deployment.

Step-by-Step Process:- Data Loading & Preprocessing

- Loaded dataset_phishing.csv.
- Dropped irrelevant columns (url) and encoded target (status).
- Performed train-test split (70%-30%).

Model Training

- Trained multiple models:
    - Logistic Regression
    - Decision Tree
    - Random Forest
- Tuned parameters where needed (e.g., max_iter=1000 for logistic regression).

Evaluation

- Metrics computed:
    - **Accuracy**
    - **Precision**
    - **Recall**
    - **F1-Score**
    - **ROC-AUC**
- Plotted bar chart of model metrics for comparison.

Model Selection

- **Random Forest** chosen based on:
    - Best F1-score
    - Best ROC-AUC
    - Strong balance of precision and recall

Model Interpretation

- Feature importance analysis highlighted top predictive features:
    - num_dots_in_url
    - url_length
    - has_https
- SHAP analysis confirmed these features heavily influence predictions.

Key Insights

- Phishing URLs often have more dots and longer length, lack HTTPS.
- Random Forest provides robust and interpretable predictions.
- The model can support business needs for automated phishing detection in security tools.

CODE (Juypter NoteBook)

```
# 1. Imports

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score

import matplotlib.pyplot as plt

import seaborn as sns

import shap


# 2. Load Data

df = pd.read_csv('dataset_phishing.csv')

X = df.drop(columns=['url', 'status'])

y = LabelEncoder().fit_transform(df['status'])


# 3. Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)


# 4. Models

models = {

  'Logistic Regression': LogisticRegression(max_iter=1000),

  'Decision Tree': DecisionTreeClassifier(random_state=42),

  'Random Forest': RandomForestClassifier(random_state=42)

}


results = []


# 5. Train + Evaluate

for name, model in models.items():

  model.fit(X_train, y_train)

  y_pred = model.predict(X_test)

  y_prob = model.predict_proba(X_test)[:,1]

  results.append({

    'Model': name,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred),
        'Recall': recall_score(y_test, y_pred),
        'F1-Score': f1_score(y_test, y_pred),
        'ROC-AUC': roc_auc_score(y_test, y_prob)
    })

# 6. Results Table + Plot
results_df = pd.DataFrame(results)
print(results_df)
results_df.set_index('Model').plot(kind='bar', figsize=(10,6))
plt.title("Model Performance Comparison")
plt.ylabel("Score")
plt.ylim(0,1)
plt.show()

# 7. Feature Importance (Random Forest)
rf = models['Random Forest']
feat_imp = pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf.feature_importances_
}).sort_values(by='Importance', ascending=False)

print(feat_imp.head(10))
sns.barplot(x='Importance', y='Feature', data=feat_imp.head(10))
plt.title("Top 10 Feature Importances")
plt.show()

# 8. SHAP
explainer = shap.TreeExplainer(rf)
shap_values = explainer.shap_values(X_test)
```

```
shap.summary_plot(shap_values[1], X_test, plot_type="bar")

shap.summary_plot(shap_values[1], X_test)
```